The Search for Computational Intelligence

Joseph Corneli [1] and **Ewen Maclean** [2]

Abstract. We define and explore in simulation several rules for the local evolution of generative rules for 1D and 2D cellular automata. Our implementation uses strategies from conceptual blending. We discuss potential applications to modelling social dynamics.

1 Introduction

This paper takes a local approach to studying the evolution of cellular automata, following on the global approach of "PICARD" [23].

> *Like a traditional one-dimensional CA, PICARD executions move from one iteration to another by some rule. However, whereas traditional CA's require the rule to be static and externally specified, PICARD infers the iteration rule from the current state of the CA itself.* [23, pp. 1–2]

PICARD's inferred rule is derived from the current state of the CA by a global characteristics, such as the number of 1's in the CA's current state (modulo 256), or the density ρ of 1's (normalised as $\rho/256$). These global criteria are similar to Van Valen's theory of resource density as an "incompressible gel" [26].

In the current paper we introduce the notion of a MetaCA, in which CA rules are derived locally at each cell within the CA as it runs. Examples appear in Figure 1. Here, each colour represents one of the 256 standard one-dimensional CA rules. States evolve locally, according to globally-defined dynamics.

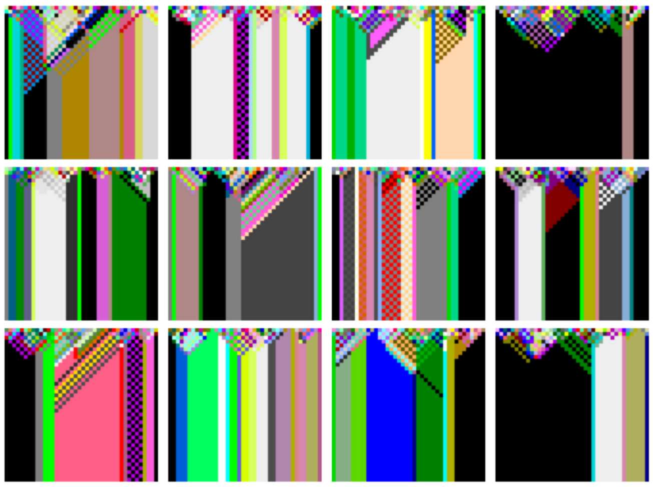


Figure 1. An illustration of MetaCA evolution

[1] Department of Computing, Goldsmiths College, University of London ✉j.corneli@gold.ac.uk
[2] School of Informatics, University of Edinburgh ✉ewenmaclean@gmail.com

2 Background

2.1 Cellular Automata

A crucial development in the history of CA research was the proof [4] that certain CA rules are Turing complete (in particular, Rule 110 in Wolfram's numbering system [28] enjoys this property).

Earlier classic works [13, 16, 22] exploring related "edge of chaos" effects. In [22, 16, 15], genetic algorithms are used to search the space of CA rules via crossover and mutation. This sort of evolution is global and is connected with the CA rule by a derived parameter, "Langton's λ" (cf. [13]). An overview of the "EvCA" programme is presented in [11].

Closest to the work presented here is [25], which introduces the paradigm of *cellular programming*. As the name indicates, this approach is a fusion of ideas from cellular automata and genetic programming.

> *As opposed to the standard genetic algorithm, where a population of independent problem solutions globally evolves, our approach involves a grid of rules that coevolves locally.* [25, p. 74]

In cellular programming, local evolution of the CA rule makes use of a local "fitness" ([25, pp. 79–81]), as the systems are evolved to perform certain global computational tasks.

In the current effort, although we are interested in behaviour that tends towards edge-of-chaos effects, system evolution is not directly guided by a specific fitness criterion, but only by variations on the "crossover" mechanism.

One early application of cellular programming was to evolutionary game theory, a field with natural parallels (cf. [21]). We will not consider game theoretic approaches in this work, despite being inspired by the social metaphors that are involved (e.g. [20]). In our thinking we often switch between conceptual/symbolic, social/ethical, biological/genetic, and physical/geometric metaphors.

In this connection it is worth mentioning some recent work [7, 8] that continues in the earlier tradition of the EvCA project (cf. [12]), making use of a relativistic "light cone" analysis to identify structure in CAs. The current paper does not pursue any detailed *post hoc* analysis of CA behaviour, although we plan to explore this further in subsequent work. Finally, although not focusing on CAs per se, [10] outlines a set of criteria for the design of systems that exhibit "emergent" intelligence which helped to motivate the present effort.

2.2 Conceptual Blending

One of our inspirations for working with cellular automata is that we are involved with a research project that studies computational blending [24], and cellular automata seem to offer a very simple example of blending behaviour. That is, they consider the value of neighbouring cells, and produce a result that “combines” these results (in some suitably abstract sense) in order to produce the next generation. We were also inspired by the idea of “blending” ordered and chaotic behaviour to produce edge-of-chaos effects.

We propose to exploit existing formalisms of blending (in the style of Goguen [9]) in the context of cellular automata to investigate emergent and novel behaviours. The fundamental building blocks used in calculating concept or theory blends are:

Input Concepts are the concepts or theories which are understood have some degree of commonality (syntactic or semantic).

Signature Morphism is a definition of how symbols are mapped between theories or concepts.

Generic Space is the space which contains a theory which is common to both input theories.

Blend is the space computed by combining both theories. The computation is computed using a “pushout” from the underlying categorical semantics [17].

Once a blend has been computed, it may represent a concept which is in some way inconsistent. Equally it may represent a concept which is in some way incomplete. We can then either weaken an input theory, or refine the blend:

Weakening Given an inconsistent blend it is possible to weaken the input concept in order to produce a consistent blend. Weakening means removing symbols or axioms from the input concept.

Refinement Given a blend which represents a concept which is in some way incomplete, it is possible to refine the concept by adding symbols or axioms.

This paper presents several examples of simple concepts to which the blending process applies. In general the notion of a signature morphism allows input concepts expressed in different languages to be blended. In this paper the examples shown have input concepts expressed in the same language, and indeed have the same specification. This means that the morphisms are not interesting and the calculated pushout could be computed without utilising the full machinery of category theory. Planned extensions will explore the idea of combining rules for cellular automata which may have entirely different techniques for expressing propagation. For this reason, we target the Heterogeneous Tool Set (HETS) system [18] as an infrastructure for computing blends. We describe our current approach to blending in the context of cellular automata in Sections 3.2 and 3.3.

3 Implementation

3.1 Generating Genotypes

Each elementary CA rule defines a mapping from all eight strings of 0’s and 1’s to the set {0,1}. Thus, for example the rule **01010100** is defined as the following operation:

$$\begin{array}{cccccc}
0 & 0 & 0 & \mapsto & \mathbf{0} \\
0 & 0 & 1 & \mapsto & \mathbf{1} \\
0 & 1 & 0 & \mapsto & \mathbf{0} \\
0 & 1 & 1 & \mapsto & \mathbf{1} \\
1 & 0 & 0 & \mapsto & \mathbf{0} \\
1 & 0 & 1 & \mapsto & \mathbf{1} \\
1 & 1 & 0 & \mapsto & \mathbf{0} \\
1 & 1 & 1 & \mapsto & \mathbf{0}
\end{array}$$

There are 256 of these rules; the example above is Rule 84 in Wolfram’s standard enumeration of 1D CAs [28]. The basic concept of the MetaCA is to evolve a CA with 256 possible states – rather than the traditional two – where each state now corresponds to a “CA rule”. Then we can then apply this rule to decide the output for the next cell, depending also on the state of the neighbouring cells. By positioning three CA rules next to each other, we define a multiplication by applying the central rule bitwise across the alleles. For example, here is the result of “multiplying” $01101110 \times 01010100 \times 01010101$. In the context of such an operation, we refer to the central term as the “local rule,” and we highlight it in bold below.

$$\begin{array}{ccccccl}
0 & \mathbf{0} & 0 & & 0 & & \textit{Apply local rule to “000”} \\
1 & \mathbf{1} & 1 & & 0 & & \textit{Apply local rule to “111} \\
1 & \mathbf{0} & 0 & & 0 & & \textit{Apply local rule to “100”} \\
0 & \mathbf{1} & 1 & \mapsto & 1 & & \textit{Apply local rule to “011”} \\
1 & \mathbf{0} & 0 & & 0 & & \textit{Apply local rule to “100”} \\
1 & \mathbf{1} & 1 & & 0 & & \textit{Apply local rule to “111”} \\
1 & \mathbf{0} & 0 & & 0 & & \textit{Apply local rule to “100”} \\
0 & \mathbf{0} & 1 & & 1 & & \textit{Apply local rule to “001”}
\end{array}$$

Realised in a simulation with random starting conditions, the results of this operation are not particularly impressive: they stabilise early and do not produce any interesting patterns (Figure 2).

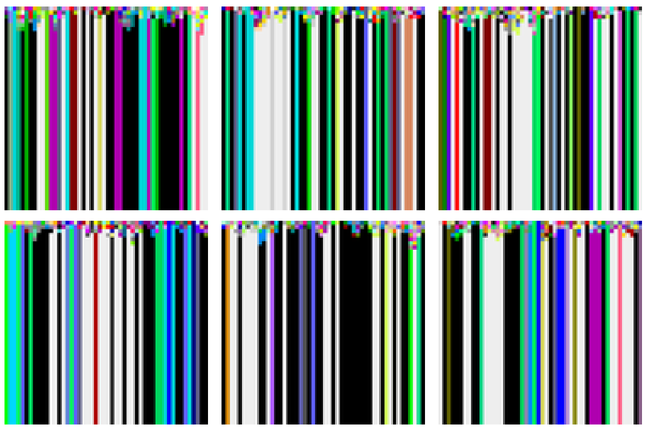


Figure 2. Under evolution according to the local rule without blending dynamics, a barcode-like stable pattern forms quickly

3.2 Introducing Blending

The blending variant says to first compute the “generic space” by noting the alleles where the two adjacent neighbours are the same, and where they differ. Only when the generic space retains some ambiguity (indicated by $\{0,1\}$) do we apply the local rule (again recorded on the centre cell at left and highlighted in bold) in a bitwise manner across each allele, to arrive at the final result.

$$\begin{array}{cccccccl}
0 & \mathbf{0} & 0 & & 0 & & 0 & \textit{Neighbours are both 0} \\
1 & \mathbf{1} & 1 & & 1 & & 1 & \textit{Neighbours are both 1} \\
1 & \mathbf{0} & 0 & & \{0,1\} & & \boxed{0} & \textit{Apply local rule to “100”} \\
0 & \mathbf{1} & 1 & \mapsto & \{0,1\} & \mapsto & \boxed{1} & \textit{Apply local rule to “011”} \\
1 & \mathbf{0} & 0 & & \{0,1\} & & \boxed{0} & \textit{Apply local rule to “100”} \\
1 & \mathbf{1} & 1 & & 1 & & 1 & \textit{Neighbours are both 1} \\
1 & \mathbf{0} & 0 & & \{0,1\} & & \boxed{0} & \textit{Apply local rule to “100”} \\
0 & \mathbf{0} & 1 & & \{0,1\} & & \boxed{1} & \textit{Apply local rule to “001”}
\end{array}$$

For illustrative purposes, this blend has been formalised in the HETS system by introducing CASL files to represent the 8 bit encodings (Listing 1, and corresponding development graph shown in Figure 3).

The computed blend is inconsistent as there is not a unique value representing the output value of each function. In order to resolve this we weaken the input rules in CASL by removing the function values which cause conflict. Note that purposes of efficiency, we have implemented our 1D experiments in LISP rather than in HETS/CASL. We've put the working code on Github[3].

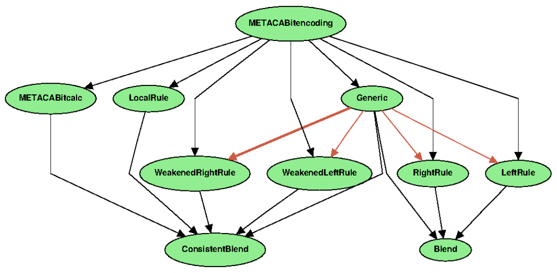
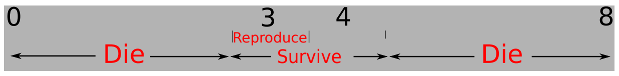


Figure 3. The development graph for calculating a blend of 8 bit encodings

3.3 2D Experiments

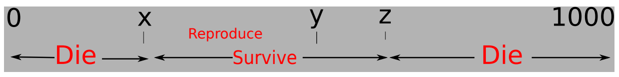
In order to extend the ideas presented so far in the 1D case, let us consider a variant of Conway's Game of Life [5], in which a global rule exists defining whether a square is alive or dead. We extend this by introducing the notion of a local rule at each square – a genotype, which governs the propagation of the phenotype.

In Conway's Game of life, one can view the rules for propagation as partitions on a finite interval $[0, 8]$.



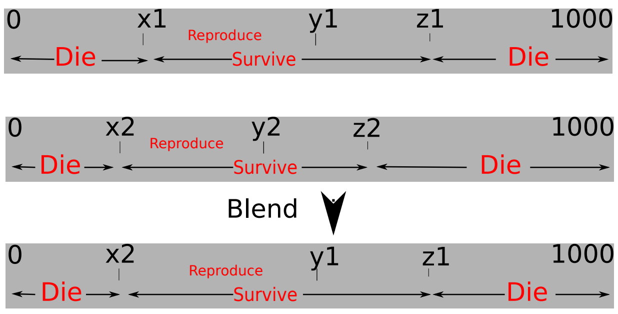
The number on the line corresponds to the number of alive neighbours adjacent, in cardinal and inter-cardinal directions, to a given square. If the square is dead then it becomes alive (labelled reproduce) if the number of alive neighbours is exactly three. If there are five or more alive neighbours the square dies from overcrowding. If there are fewer than three alive neighbours the square dies from underpopulation. In all other cases the square maintains its status.

This can be generalised to partitions within a more finely grained line, for example from 0 to 1000, one creates a genotype (x, y, z):



We introduce the corresponding notion of a *weight* for each cell. The *phenotype* of the cell is then a pair $(alive, weight)$ which denotes whether the cell is alive, and what weight is has. In this paper we always calculate a newly propagated weight as the average of the neighbours' weights.

The notion of local propagation is introduced by allowing the genotypes to be blended at each point where a cell remains or becomes alive. As we have represented the genotype as a partitioned line, we can, for example perform a blend where the partition is blended in such a way as the minimise the lowest bound and maximise the highest bound, and maximise the interval for reproduction. Given two genotypes (x_1, y_1, z_1) and (x_2, y_2, z_2), the blend is $(\min\{x_1, x_2\}, \max\{y_1, y_2\}, \max\{z_1, z_2\})$



Note that this is just one of several possible blending strategies, which we refer to as a *union* blend, since it maximises the partitions which pertain to survival. We consider alternative blends in our experiments.

4 Results

4.1 1D CAs

One of the first things we noticed was that even though the blending dynamic creates more interesting "CA-like" patterns than simple evolution according to the local rule (as illustrated in Figure 1), it also forms stable bands after this interesting initial period. In Figure 4, this is illustrated in a CA running with 500 cells over 500 generations. Figure 4 also includes a phenotype (in black and white) which is driven entirely by the genotype: that is, if the local genotype is $\boxed{\alpha}\boxed{\beta}\boxed{\gamma}$ where $\alpha, \beta, \gamma \in \{0, 1\}^8$ and the local phenotype is $\boxed{a}\boxed{b}\boxed{c}$ where $a, b, c \in \{0, 1\}$, then the genotype evolves locally according to the meta-rule $\alpha \times \beta \times \gamma$ (in the blending variant) while the phenotype evolves by applying the local rule β to the data "*abc*."

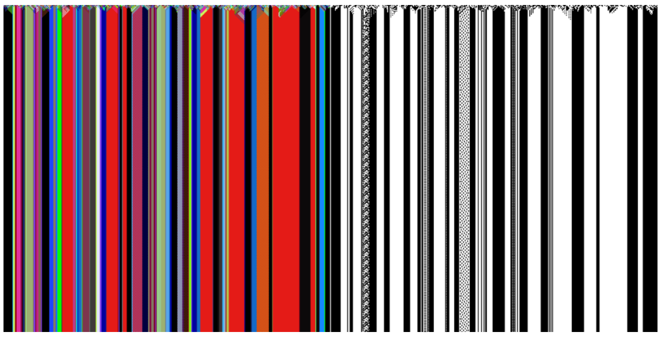


Figure 4. Phenotype with behaviour determined by genotype

In the phenotype layer, we see a few bands with interesting patterns, where the MetaCA at left has stabilised locally into one of the more interesting CA rules. However, the long term evolution is not particularly interesting: the structure observed in Figure 1 disappears quickly.

We therefore decided to introduce random mutations to the genotype, illustrated in Figures 5–7. With a high mutation rate, both genotype and phenotype are almost reduced to confetti. If we reduce the mutation rate sufficiently, some degree of stability is preserved, and the vertically striped bands are transformed into intermingling swaths

[3] `https://github.com/holtzermann17/metaca`

of colour (Figure 6). We also see areas with more finely-grained structure in the phenotype layer.

In Figure 7, the colour-coded genotype layer has been replaced with a greyscale coding, and we see more clearly how the phenotype behaviour follows that of the genotype. That is, genotypes similar to Rule 0 (00000000) or Rule 256 (11111111) tend to produce 0 or 1, respectively, in the phenotype layer. Rules that output a blend of 0's and 1's are mapped to grey shades. Several interesting rules (Rule 110, Rule 30, Rule 90, Rule 184, and their reversals, bitwise inverses, and inverted-reversals) are highlighted in colour. In particular, Rule 110 variants are highlighted in red.

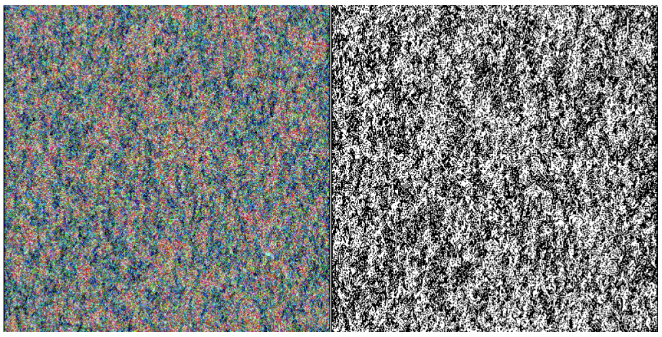


Figure 5. A high rate of mutation produces tantalising random structures

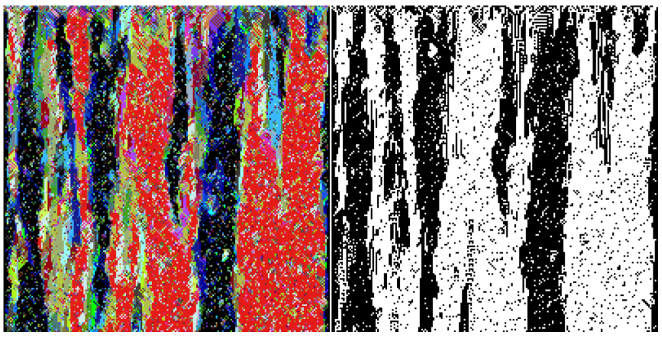


Figure 6. Throttling down the mutation rate preserves some of the large-scale stability while making room for variability

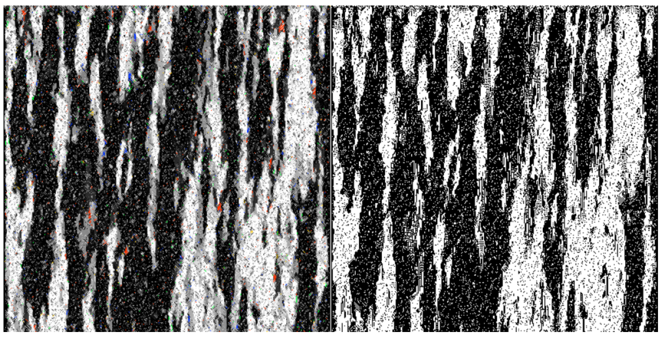


Figure 7. The search for intelligent life in the computational universe

We observe that Rule 0 and Rule 256 behaviour tends to predominate. Grey areas appear to be semi-stable. Red patches appear and disappear, as if independent planets evolve intelligent life and are then extinguished. With this physics, "intelligent life" seems inevitable, but also inevitably short-lived. One would have to look for another overall physics for intelligent behaviour to predominate.

A potential indication of the direction to look in is presented in Figure 8, which presents CAs generated by adjusting the typical blending evolution pattern by an (erroneously-programmed) mutation rule that only flips the first bit. We see that long-term behaviour in the genotype flutters randomly between Rule 0 (00000000) and Rule 128 (10000000). The short-term behaviour in the phenotype is nevertheless quite interesting, exhibiting many of the familiar lifelike edge-of-chaos patterns before ultimately succumbing to a version of Newton's First Law.

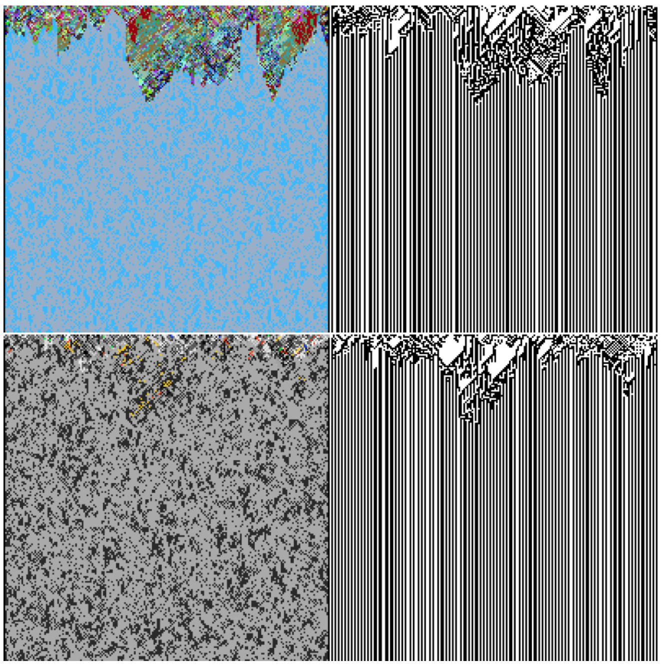
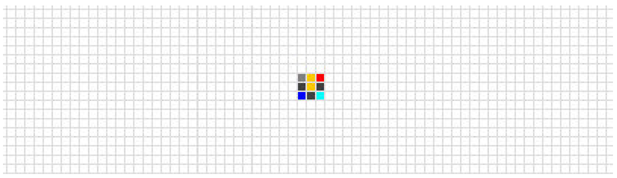


Figure 8. A skewed mutation pattern

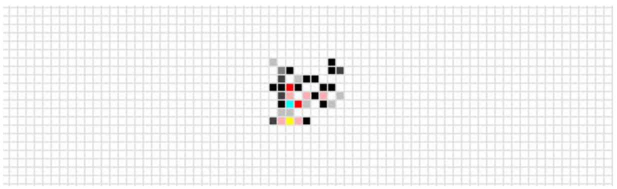
4.2 2D CAs

To see the behaviour of the union blend in action consider an initially populated grid, where colours represent the weights of alive cells:

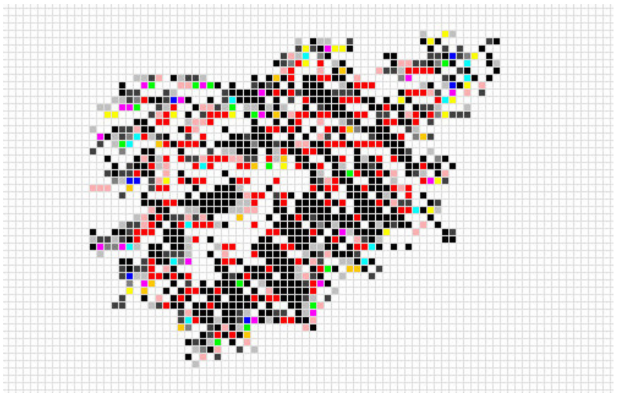


For this example, we initially restrict the computation of the blend for a particular cell to take place when the cell is alive in the next iteration. Also we compute the blend of genotype for all neighbours, whether dead or alive.

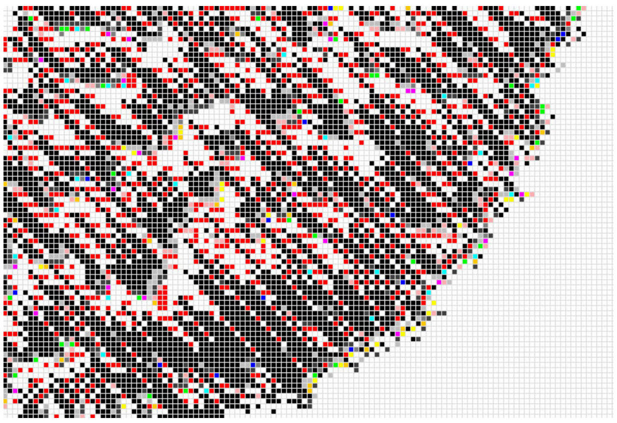
After 300 iterations the colony has grown a small amount:



Over time, the population continues to grow, with large patches of low-weight (black) cells:



Finally some structure starts to appear in the clustering:



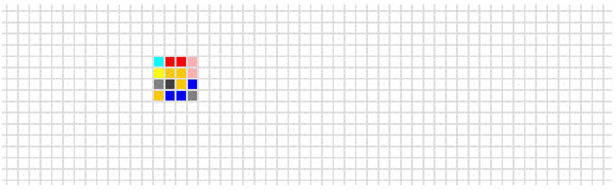
The propagation that follows shows a population of cells which grows slowly overtime. The majority of the members have low weight (represented by black squares), but interspersed within the population are chains of squares with high weight (represented by red squares) adjacent to dead cells (white).

4.2.1 *Modified Blends*

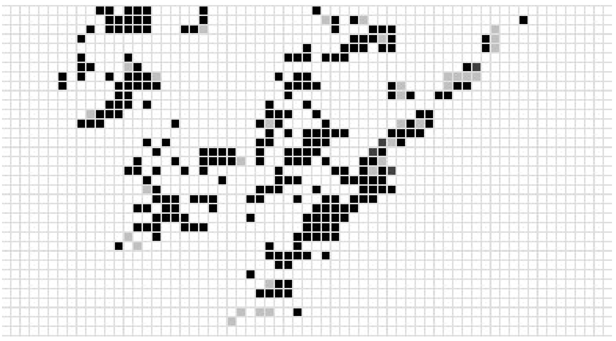
So far we have only showed the union blend working on the genotype. However, it is possible to use different blending techniques:

- Consider blending only the genotypes of alive neighbours, or all neighbours;
- Consider only blending genotypes for cells which are alive after propagation;
- Consider an *intersection* blend, where the partition sizes for survival are minimised;
- Consider an *average* blend, where the values of each genotype (x_i, y_i, z_i) are summed and divided by either the number of alive neighbours, or the total number of neighbours.

As an example of different observed emergent behaviour consider a union blend where the blend is only computed from alive neighbours, and as before we compute only for cells which are alive at the next iteration. We start with an initial state:

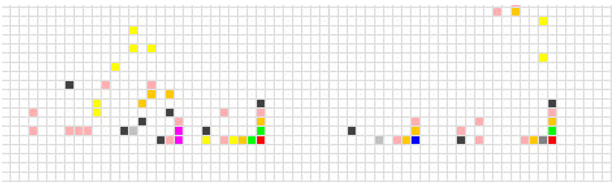


and observe a changing, but relatively steady pattern (resembling the motion of a flame) which does not grow in size using the union blend:

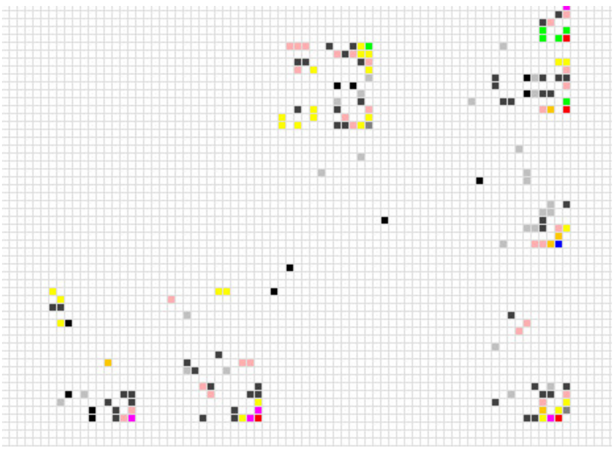


where the weight characteristic of the phenotype of each cell has fallen to very low.

Finally, consider applying instead an average blend under the same initial conditions:



Then we see a less steady but more active growth, with populations moving in triangular shapes away from population centres, leaving very small but steady and inactive populations behind:



the quickly-moving populations do not have a convergent weight characteristic of the phenotype, as in the case with the union blend for the same initial conditions.

```
library metaca

logic CASL

spec METACABITENCODING =
      free type Bit ::= 0 | 1
      sort   Triple
      ops    t : Bit × Bit × Bit → Triple;
             bitop__ : Triple → Bit
end

%% How to calculate a blend given three 8-bit genotypes
spec METACABITCALC =
      METACABITENCODING
then op     blend____ : Triple × Triple → Bit
      ∀ t1, t2, t3 : Triple
      • bitop t1 = bitop t2 ⇒ blend t1 t2 = bitop t1
      • ¬ bitop t1 = bitop t2 ⇒ blend t1 t2 = bitop t3
end

spec LEFTRULE =
      METACABITENCODING
then • bitop t(0, 0, 0) = 0
      • bitop t(0, 0, 1) = 1
      • bitop t(0, 1, 0) = 1
      • bitop t(0, 1, 1) = 0
      • bitop t(1, 0, 0) = 1
      • bitop t(1, 0, 1) = 1
      • bitop t(1, 1, 0) = 1
      • bitop t(1, 1, 1) = 0
end

spec RIGHTRULE =
      METACABITENCODING
then • bitop t(0, 0, 0) = 0
      • bitop t(0, 0, 1) = 1
      • bitop t(0, 1, 0) = 0
      • bitop t(0, 1, 1) = 1
      • bitop t(1, 0, 0) = 0
      • bitop t(1, 0, 1) = 1
      • bitop t(1, 1, 0) = 0
      • bitop t(1, 1, 1) = 1
end

spec LOCALRULE =
      METACABITENCODING
then • bitop t(0, 0, 0) = 0
      • bitop t(0, 0, 1) = 1
      • bitop t(0, 1, 0) = 0
      • bitop t(0, 1, 1) = 1
      • bitop t(1, 0, 0) = 0
      • bitop t(1, 0, 1) = 1
      • bitop t(1, 1, 0) = 0
      • bitop t(1, 1, 1) = 0
end

%% Generic is common between left and right
spec GENERIC =
      METACABITENCODING
then • bitop t(0, 0, 0) = 0
      • bitop t(0, 0, 1) = 1
      • bitop t(1, 0, 1) = 1
end

%% Morphism from Generic to Left
view LEFT : GENERIC to LEFTRULE
end

%% Morphism from Generic to Right
view RIGHT : GENERIC to RIGHTRULE
end

%% This will be inconsistent
spec BLEND =
      combine Left, Right
end

spec WEAKENEDLEFTRULE =
      METACABITENCODING
then • bitop t(0, 0, 0) = 0
      • bitop t(0, 0, 1) = 1
      • bitop t(0, 1, 0) = 1
      • bitop t(0, 1, 1) = 0
      • bitop t(1, 0, 0) = 1
      • bitop t(1, 0, 1) = 1
      • bitop t(1, 1, 0) = 1
end

spec WEAKENEDRIGHTRULE =
      METACABITENCODING
then • bitop t(0, 0, 0) = 0
      • bitop t(0, 0, 1) = 1
      • bitop t(1, 0, 1) = 1
      • bitop t(1, 1, 1) = 1
end
view WEAKENEDLEFT : GENERIC to WEAKENEDLEFTRULE
end
view WEAKENEDRIGHT : GENERIC to WEAKENEDRIGHTRULE
end

%% A computed blend as new 8 bit encoding
spec CONSISTENTBLEND =
      combine WeakenedLeft, WeakenedRight
and   METACABITCALC
and   LOCALRULE
end
```

Listing 1. CASL source code listing calculating the running example $01101110 \times 01010100 \times 01010101$ via the blending meta-rule

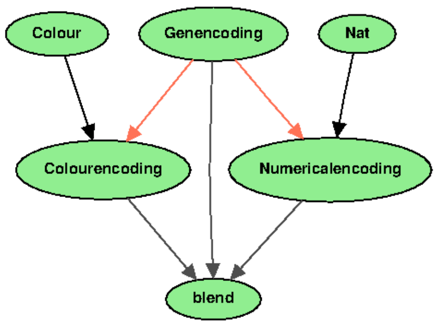


Figure 9. Blending different 2d genotypes

```
library metaca2d

logic CASL

spec NAT =
      sort   Nat
      op     max : Nat × Nat → Nat
      op     min : Nat × Nat → Nat
end

spec COLOUR =
      sort   Colour
      op     maxhue : Colour × Colour → Colour
end

%% a 2−d cellular automaton with numerical Genotype
spec NUMERICALENCODING =
      NAT
then sort   NGenotype
      ops    genotype : Nat × Nat × Nat → NGenotype;
             t : Nat × Nat × Nat → NGenotype;
             numblend : NGenotype × NGenotype → NGenotype
      ∀ g1, g2 : NGenotype; x1, y1, z1, x2, y2, z2, x3, y3, z3 : Nat
      • g1 = t(x1, y1, z1) ∧ g2 = t(x2, y2, z2)
        ⇒ numblend(g1, g2)
           = t(min(x1, x2), min(y1, y2), max(z1, z2))
end

%% A colour CA Genotype
spec COLOURENCODING =
      COLOUR
then sort   CGenotype = Colour
      op     hueblend : CGenotype × CGenotype → CGenotype
      ∀ g1, g2 : CGenotype
      • hueblend(g1, g2) = maxhue(g1 as Colour, g2 as Colour)
end

%% A generic space
spec GENENCODING =
      sort   S
      sort   Genotype
      op     blend : Genotype × Genotype → Genotype
end

%% A signature morphism from Generic to Numerical
view NUMERICALSM :
      GENENCODING to NUMERICALENCODING =
      S ↦ Nat, Genotype ↦ NGenotype, blend ↦ numblend
end

%% A signature morphism from Generic to Colour
view COLOURSM :
      GENENCODING to COLOURENCODING =
      S ↦ Colour, Genotype ↦ CGenotype, blend ↦ hueblend
end

spec BLEND =
      combine NumericalSM, ColourSM
end
```

Listing 2. CASL source code using signature morphisms and pushout calculation to blend genotypes with different languages

5 Discussion

5.1 Research Contribution

The motivation for combining a notion of blending with cellular automata was to investigate ways in which cellular automata could be used to model processes, where propagation rules, or genotypes, were locally defined. The main research contributions in the field of two dimensional cellular automata are

- We built and implemented a framework where local propagation experiments can be performed;
- We used the HETS system to show that the notion of blending can be used to invent new propagation rules for different genotypes;
- We invented simply definable genotypes and blends of these genotypes to show proof of concept;
- Finally, we shared the results of simulations that illustrate qualitative behaviour in one and two dimensional MetaCAs.

The primary limitation of this work is that our results are purely observational at present. For example, the early experiments seemed to provide visual evidence that blending is useful: Figure 1 more interesting than Figure 2. The robustness of our qualitative findings have been supported by developing a range of different experiments, for example, some analogy could be drawn between the "grey areas" observed in Figure 7 for the 1D case and the red-and-white chains that develop in the 2D case under union blending.

Our results confirm the basic finding of CA research: interesting global behaviour can arise from simple rules governing local interactions, with the added twist the rules can also arise locally. The MetaCA setting seems to offer fertile ground for further computational research into evolutionary and co-evolutionary effects.

5.2 Social Interpretation

One can view the propagation of cells and patterns in a 1D or 2D MetaCA as a social process, and blending as a knowledge exchange. In the 2D case, we can think of the generated diagrams as illustrations of interactions between individuals with high knowledge, skill, or social impact (high weight), and those with less (low weight). The propagation in the "union" blend shows how large numbers of individuals with low social impact outnumber those with high social impact, but those with high social impact impose the emergent structure and determine the growth of the group of individuals.

In a fundamental respect our blending rules seem to embody a thought-provoking blend of two very different kinds of "ethics." Specifically: blending seems to introduce a dynamic similar to Carol Gilligan's *ethic of care* [6], which seeks to defend the relationships that obtain in a given situation. Here this is manifested by the question "Have my neighbours already formed a consensus?" This behaviour augments and extends the local rule, which would correspond to Lawrence Kohlberg's *ethic of justice* (cf. [2]).

As we saw in Section 4.1, we would have to work harder to find meta-rules that give rise to an "intelligent universe" or in which life (considered as symbolic computation) plays an obvious negentropic role (*après* Bergson [3]).

One strategy that has not been developed here would be to make use of a "Baldwin effect" [1, 27], to use "learning" (considered as entropy) in the phenotype layer to drive evolution. More specifically, $\boxed{0}\boxed{0}\boxed{0} \mapsto \boxed{0}$ and $\boxed{1}\boxed{1}\boxed{1} \mapsto \boxed{1}$ seem to be relatively uninteresting behaviours, but they are also hard to resist under the blending dynamics as we've defined them (compare Figures 4 and 7). Can we find ways to select against them?

5.3 Planned extensions

One observes that under our blending rule, the two non-entropic behaviours listed above tend to selected for, not against, because they are examples of the "neighbours match" condition. Indeed, reviewing the essential features of blending in the 1D case, we can use our basic principles:

"*If neighbours match: use their shared value as the result.*
If neighbours don't match: use local logic to get the result."

to define a 1D CA rule, if we interpret "local logic" to mean "substitute my own value as the result." Here's how we would then define blending for triplets:

0	0	0	$\mapsto$	0	*Neighbours match*
0	0	1	$\mapsto$	0	*Local logic*
0	1	0	$\mapsto$	0	*Neighbours match*
0	1	1	$\mapsto$	1	*Local logic*
1	0	0	$\mapsto$	0	*Local logic*
1	0	1	$\mapsto$	1	*Neighbours match*
1	1	0	$\mapsto$	1	*Local logic*
1	1	1	$\mapsto$	1	*Neighbours match*

This is Wolfram's Rule 23: and as it happens, its evolutionary behaviour is not particularly interesting. Of course, for blending at the genotype level, "local logic" can be determined by any CA. Even so, when we use blending bitwise on alleles, we only ever run the local logic on half of the cases, and moreover it always the same half, determined by a "censored" version of Rule 23.

0	0	0	$\mapsto$	0	*Neighbours match*
0	0	1	$\mapsto$	$*$	*Local logic*
0	1	0	$\mapsto$	0	*Neighbours match*
0	1	1	$\mapsto$	$*$	*Local logic*
1	0	0	$\mapsto$	$*$	*Local logic*
1	0	1	$\mapsto$	1	*Neighbours match*
1	1	0	$\mapsto$	$*$	*Local logic*
1	1	1	$\mapsto$	1	*Neighbours match*

Rather than using Censored Rule 23 as our template, we could instead have the template determined by phenotype data, thereby inserting the phenotype as a "hidden layer" in the computation.

The standard template could be understood to be generated by locking in $\boxed{0}\boxed{0}\boxed{0} \mapsto \boxed{0}$ along with a "variation"[4] $\boxed{0}\boxed{1}\boxed{0} \mapsto \boxed{0}$ and the bitwise inverses of these. A wider class of templates could calculated from arbitrary phenotype data by the same operations. What we would lose in abandoning the intuition associated with local blending, we may be repaid through a much more abstract but richer procedural blend, operating at the level of genotype+phenotype evolution. At the very least, we can point to a generic space, namely the locked-in local rule which would be carried over (along with its variants) from the phenotype to the corresponding alleles.

As a simple example of cross-domain blending consider a genotype defined as in §3.3, and another which is defined by comparing the hue of just one neighbour. Their blend is a richer theory combining elements from both genotypes. CASL code expressing these concepts is given in Listing 2, and the resulting categorical diagram can be seen in Figure 9. Experimentation with more sophisticated genotypes and blends is ongoing.

[4] $\boxed{0}\boxed{1}\boxed{0} = \boxed{0}\boxed{0}\boxed{0} + \boxed{}\boxed{1}\boxed{}$

5.4 Future work

Coevolution has been understood to be relevant from both a philosophical [14] and empirical perspective [26]. Finding patterns that allow us to exploit Baldwin effects to drive the co-evolution of genotype and phenotype in the direction of intelligent behaviour is an interesting computational project. The MetaCA domain may help to show how to systematise some aspects of the search for the principles and techniques that underlie broader computational intelligence.

Expanding on the relatively simple domain of CAs, we would like to use HETS to formalise the mechanisms of social knowledge sharing and problem solving in fields like mathematics. Indeed it may be possible in the future to encode mathematical problems in a MetaCA or cellular program and see how a group of agents can solve the problems as a society. This would be informed by ongoing empirical analysis of real problem-solving activities [19] developed in parallel to the simulation work presented here.

6 Conclusion

This research was inspired by the aim to build an example of computational blending that matched, to some extent, the way blending might work in social settings. One person suggests an idea, and another offers a variant of that, a third brings in another idea from elsewhere and some combination is made. The next day, things head in another direction completely. Our progress in this research project has followed this sort of trajectory: from an initial critique of blending theory ("it's not dynamic enough to be social!") to some tentative examples showing how large-scale system dynamics can be driven by local behaviour in an emergent manner. Perhaps the most interesting aspect of this research is the relationship between these emergent dynamics and the meta-rules. Whereas previous CA research has shown that complex global behaviour can be generated from a set of simple, local rules, this project gives an enticing glimpse of a future research programme that carries out a computational search for those very rules (out of the many possible) that lead to system behaviour we would recognise as "intelligent."

7 Acknowledgements

This research has been funded by the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open Grant number 611553 (COINVENT).

We thank Timothy Teravainen, Raymond Puzio, and Cameron Smith for helpful conversations and pointers to literature.

REFERENCES

[1] James Mark Baldwin, 'A New Factor in Evolution', *American Naturalist*, **30**, 441–451, 536–553, (1896).

[2] Seyla Benhabib, 'The Generalized and the Concrete Other: The Kohlberg-Gilligan Controversy and Feminist Theory', *PRAXIS International*, **4**, 402–424, (1985).

[3] Henri Bergson, *Creative evolution*, University Press of America, 1912 [1907].

[4] Matthew Cook, 'Universality in elementary cellular automata', *Complex Systems*, **15**(1), 1–40, (2004).

[5] M. Gardner, 'The fantastic combinations of John Conway's new solitaire game "life"', *Scientific American*, **223**, 120–123, (October 1970).

[6] Carol Gilligan, *In a different voice*, Harvard University Press, 1982.

[7] Georg M Goerg and Cosma Rohilla Shalizi, 'LICORS: Light cone reconstruction of states for non-parametric forecasting of spatio-temporal systems', *arXiv preprint arXiv:1206.2398*, (2012).

[8] Georg M Goerg and Cosma Rohilla Shalizi, 'Mixed LICORS: A Nonparametric Algorithm for Predictive State Reconstruction', *arXiv preprint arXiv:1211.3760*, (2012).

[9] Joseph Goguen, 'Mathematical models of cognitive space and time', in *Reasoning and Cognition: Proceedings of the Interdisciplinary Conference on Reasoning and Cognition*, eds., D. Andler, Y. Ogawa, M. Okada, and S.Watanabe, pp. 125–148, Tokyo, (2006). Keio University Press. On-line version updated.

[10] Douglas Hofstadter, 'Prolegomena to any future Metacat', in *Fluid concepts and creative analogies*, pp. 307–318. Basic Books, Inc., (1995).

[11] Wim Hordijk, 'The EvCA project: A brief history', *Complexity*, **18**(5), 15–19, (2013).

[12] Wim Hordijk, Cosma Rohilla Shalizi, and James P Crutchfield, 'Upper bound on the products of particle interactions in cellular automata', *Physica D: Nonlinear Phenomena*, **154**(3), 240–258, (2001).

[13] Chris G Langton, 'Computation at the edge of chaos: phase transitions and emergent computation', *Physica D: Nonlinear Phenomena*, **42**(1), 12–37, (1990).

[14] George Herbert Mead, *The philosophy of the present*, Open Court, 1932.

[15] Melanie Mitchell, James P Crutchfield, and Peter T Hraber, 'Evolving cellular automata to perform computations: Mechanisms and impediments', *Physica D: Nonlinear Phenomena*, **75**(1), 361–391, (1994).

[16] Melanie Mitchell, Peter Hraber, and James P Crutchfield, 'Revisiting the edge of chaos: Evolving cellular automata to perform computations', *Complex Systems*, **7**, 89–130, (1993).

[17] Till Mossakowski, Christian Maeder, and Klaus Lüttich, 'The Heterogeneous Tool Set', in *TACAS 2007*, eds., Orna Grumberg and Michael Huth, volume 4424 of *Lecture Notes in Computer Science*, pp. 519–522. Springer-Verlag Heidelberg, (2007).

[18] Till Mossakowski, Christian Maeder, and Klaus Lüttich, 'The heterogeneous tool set, HETS', in *Tools and Algorithms for the Construction and Analysis of Systems*, eds., Orna Grumberg and Michael Huth, 519–522, Springer, (2007).

[19] Dave Murray-Rust, Joseph Corneli, Alison Pease, Ursula Martin, and Mark Snaith, 'Synchronised multi-perspective analysis of online mathematical argument', in *Proceedings of 1st European Conference on Argumentation: Argumentation and Reasoned Action, 9-12 June 2015, Lisbon, Portugal*, eds., Sally Jackson, Dima Mohammed, Lilian Bermejo-Luque, and Steve Oswald, (2015). To appear.

[20] M.A. Nowak, 'Five rules for the evolution of cooperation', *Science*, **314**(5805), 1560–1563, (2006).

[21] M.A. Nowak and R.M. May, 'Evolutionary games and spatial chaos', *Nature*, **359**(6398), 826–829, (1992).

[22] Norman H Packard, 'Adaptation toward the edge of chaos', in *Dynamic Patterns in Complex Systems*, eds., J. A. S. Kelso, A. J. Mandell, and M. F Shlesinger, 293–301, World Scientific, (1988).

[23] Theodore P Pavlic, Alyssa M Adams, Paul CW Davies, and Sara Imari Walker, 'Self-referencing cellular automata: A model of the evolution of information control in biological systems', in *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, 522–529, The MIT Press, (2014).

[24] Marco Schorlemmer, Alan Smaill, Kai-Uwe Kühnberger, Oliver Kutz, Simon Colton, Emilios Cambouropoulos, and Alison Pease, 'COINVENT: towards a computational concept invention theory', in *Proceedings of the 5th International Conference on Computational Creativity*, eds., Dan Ventura, Simon Colton, Nada Lavrac, and Michael Cook, (2014).

[25] Moshe Sipper, *Evolution of parallel cellular machines*, Springer Heidelberg, 1997.

[26] Leigh Van Valen, 'A new evolutionary law', *Evolutionary theory*, **1**, 1–30, (1973).

[27] *Evolution and learning: The Baldwin effect reconsidered*, eds., Bruce H Weber and David J Depew, MIT Press, 2003.

[28] Stephen Wolfram, *Cellular automata and complexity: Collected papers*, Addison-Wesley Reading, 1994.